BOY MEETS GOAL, BOY LOSES GOAL, BOY GETS GOAL: The nature of feedback between goal-based simulation and understanding systems

James R. Meehan
Artificial Intelligence Project

Technical Report #170

March 1981

This work was supported in part by the Naval Ocean Systems Center under contract N66001-80-C-0377.

TABLE OF CONTENTS

Boy Meets Goal, Boy Loses Goal, Boy Gets Goal: The nature of feedback between goal-based simulation and understanding systems

James R. Meehan
Artificial Intelligence Project
Dept. of Information and Computer Science
University of California
Irvine, California 92717

ABSTRACT. We are designing a goal-based planning and simulation system called REACTOR for a multiple-actor world in which partially formulated plans are monitored during execution, providing feedback to the planner. Plan failures that occur are diagnosed by a combination of top-down (plan-synthesis) and bottom-up (plan-understanding) techniques, allowing an informed choice of response to the error. By maintaining separate belief spaces for each actor, we simulate planners who themselves simulate the planning and plan-understanding of other actors.

1.0 Introduction

Problem-solving in a world of multiple actors and uncertain information presents difficulties not apparent from studying single-actor systems, such as robot planners, that operate with complete and correct information. We are designing a program that combines both planning and plan-understanding, taking into account several aspects:

This research was supported by the Naval Oceans Systems Center under Grant N66001-80-C-0377.

1. Separate belief spaces for each actor, so that plans are formed, for example, using only that actor's information, which can be incorrect and incomplete (and nearly always is).

2. Feedback simulation, where the execution of an actor's plans and the observation of others' plans are monitored in terms of the actor's goals and beliefs.

3. Shallow planning, where an actor can formulate a plan without worrying about all the details that he would when executing that plan.

4. Detailed analysis of errors during plan execution (i.e., "What went wrong?"), followed by re-planning and adaptation.

5. Hypothetical planning, where an actor simulates some other actor's problem-solving processes. This is used as a technique for understanding the actions of others.

This theory of planning and plan-understanding is part of a natural-language system we are currently building, called REACTOR. The principal new feature of this system is its ability to monitor the execution of its plans and to modify them when necessary on the basis of their failures. There are existing language-oriented problem-solving systems that

produce actions from goals [11], and text-understanding systems that infer goals from actions [3,20], but we believe that a better model of natural-language processing incorporates both by being completely flexibile with respect to the "direction" of planning.

Consider the following story:

> John wanted to meet Mary, and he received an invitation to a party given by Jane. John believed that Jane and Mary were good friends, so he accepted the invitation. An hour into the party, John had still not seen Mary.

Viewed as a task for a natural-language understanding program, the question is what predictions will the program make for John's future actions. Viewed as a task for a problem-solving system (with a natural-language generator, perhaps), the question is what the program should simulate in terms of John's future actions. We want our theory to be relatively independent of the "direction" of the program, understanding or generation, because we believe that in either case, the most important questions are the same, namely <u>What went wrong?</u> and <u>What should John do next?</u>

2.0 Background

2.1 Plan-understanding systems

Over the last several years, natural language systems have improved in their ability to deal with story-like texts, not only because they employ more sophisticated techniques for mapping words into meaning representations, but also because they rely on new theories of human intentionality. Indeed, the "parsers" have less responsibility than they used to, because they are better integrated with inference mechanisms and knowledge structures, and much of the task of understanding lies in being able to follow the plans and goals of the actors in the text. The particular systems we will refer to are Wilensky's PAM [20] and Granger's ARTHUR [3]. These programs rely heavily on the ability to predict possible future input, at several levels of abstraction, and to instantiate or match actual future input with these predictions, generating still more predictions by means of an inference mechanism that knows about plans and goals. These programs build a representation of the causal structure of the story as understanding proceeds.

The prediction/instantiation process is not infallible: it can happen that a new sentence corresponds to no prediction, or it may even contradict the causal structure. These are not errors of programming or theory. Rather, they indicate legitimate disagreements between what is reasonable

to predict and what actually occurs. In the case of a strongly predicted or highly contradictory event, a reader might experience surprise or even express disbelief, indicating that a major revision of the explanation structure will be required. But many types of "errors" correspond to accidents that occur in the story, and human readers can often detect accidents, incorporate the structural revisions, and hardly notice that anything unusual happened.

ARTHUR corrects erroneous inferences by supplanting links in the causal structure with ones that provide an alternative explanation, even if that forces an interpretation that may seem unlikely. It takes the view that flexibility is more important than initial accuracy; if you don't get it right the first time, that's okay: you can correct it later on.

2.2 Plan-based simulators

While there are many examples of problem-solving systems, they tend to deal with single-actor micro-worlds. (For a survey of such systems, see Sacerdoti [13].) There are also programs that model the goals of discourse [6,7,8], but all of these are in general very different from the style of program we have in mind. The model on which we're basing the current work is TALE-SPIN [11], a story-generator and problem-solver for a micro-world of multiple actors,

based on the theory that stories are frequently goal-driven [2].

TALE-SPIN's planning strategies were adapted from Schank and Abelson [14], as were those in PAM and ARTHUR, yet it was not the inverse of those programs, as one might expect. In fact, their implementations varied dramatically [12,19]. But we now expect future natural language systems to integrate plan-understanding and plan-synthesizing abilities, not necessarily to the point where they are in essence different interpreters of a common data base of planning knowledge, but at least so that they can understand what they themselves generate.

3.0 An example

Here again is our story:

> John wanted to meet Mary, and he received an invitation to a party given by Jane. John believed that Jane and Mary were good friends, so he accepted the invitation. An hour into the party, John had still not seen Mary.

We will analyze the story from John's viewpoint. The background information in John's belief space is that Mary and Jane are friends and that he wants to meet Mary. The initial event is that he receives an invitation from Jane for a party. The system's knowledge of "party" and "invitation" is sufficient for John to recognize that this is a REQUEST, and that John must respond. How does he do

this? If John hates parties, he will decline. If he is an avid party-goer, he will accept. But suppose neither is the case. Then he must decide. To do this, he needs to examine the situation in more detail, which he does by imagining (mentally simulating) Jane giving a party. He can do this because he knows how one gives parties, and he generates such a situation with Jane in the role of hostess. The system knows that a party has, say, three principal activities, eating, drinking, and socializing. He can now react to each of those three. Again, there may be immediate reactions: if he's on a strict diet, or he's always interested in a free drink, he may decide right away. He might also explore each of those by considering what food Jane is likely to serve, etc., which again he would do by simulating Jane's behavior "in his head."

The decision whether to consider such details is part of what we call shallow planning. The factors include the planner's particular interests and habits, as well as his known goals, past, present, and future. These are the "top-down" considerations. But there are "bottom-up" considerations that also serve to reduce the search. Both of these kinds of considerations, or "planning filters," are organized in REACTOR by the kind of indexed memory that ARTHUR used in understanding plans [3]. They can also influence the order in which various aspects of a hypothetical plan are considered.

Suppose John has no strong reactions to the eating or drinking parts of the party. In considering the social aspect, he considers that Jane is likely to invite people she likes. He notices that since he has received an invitation, Jane probably likes him. He may react to that, especially if it is a new or contradictory piece of information. But he also notices that Jane will probably invite Mary, since he thinks Jane likes Mary. If she attends the party and he does too, then he will have achieved one of the preconditions for his goal of meeting her, namely, being in the same place. This is a very strong positive reaction, so he decides to respond to Jane's request by accepting the invitation. So far so good.

Now John is at the party and hasn't seen Mary, even though the party has been going on for an hour. The plan isn't working, and John notices that and attempts to explain the failure. He notices it because a time limit has expired for achieving the goal. This is one of several ways in which plan-failures can be noticed; most others are more direct, and correspond to the sources of plan failure, described below.

Explaining plan failure may or may not be simple. Some plans may already have attached to them a set of likely causes for failure. We will see some examples shortly. But assuming that John doesn't have a ready explanation for not yet having seen Mary, he searches for a reason by

synthesizing and examining what he believes Mary's plan is. He begins by assuming that she had a plan that worked (i.e., would have brought her to the party), and he then questions each part of that plan. In John's hypothesizing, Jane has sent Mary an invitation; Mary has received the invitation; she understands it (when the party was being held, etc.); she decides to go to the party; she goes to the party; she runs in to John. John knows that the "top" of this sequence is true (Jane is giving a party), but the "bottom" isn't (he hasn't run in to Mary). He examines the parts of this plan in detail. (For this presentation, we will look at them in bottom-up order.)

1. Problem: Mary is at the party, but I haven't run in to her.

   Explanation: She's in some other room.

   Response: I can find her by wandering around.

   Note: I've done that for the past hour, so that's not the explanation.

2. Problem: She decided to go, but she isn't here.

   Explanation 1: She isn't here yet, but she will arrive later.

   Response: There's nothing to do but wait some more.

   Explanation 2: Something is interfering with her plan to get here.

   Response: Examine her plan for getting here (e.g., D-PROX). [At this point, he synthesizes Mary's plan for getting to the party. Given what little he knows about her, he decides that she will use her car.]

   1. Problem: she decided to use her car, but it didn't work.

Explanations (immediate, since we have knowledge about car problems already stored, so we need not expand the USE-CAR plan): dead battery, no gas, flat tire.

Response: she should find a gas station.

Note: finding a gas station is not likely to be a problem, so there's no point in pursuing this path.

[Since that was the only reasonable technique for D-PROX, that path is ended.]

3. Problem: she understood the invitation, but she decided not to go.

Explanation 1 (immediate): Mary is sick. This is a pre-stored reason to turn down any invitation.

Response: none. It is very unlikely that being sick was part of her plan.

Explanation 2: Mary hates parties; i.e., she herself had an immediate, negative reaction to the idea of aoing to a party.

Response: none (unaffectable state).

Note: if this is the most likely explanation (after other possibilities have been considered), figure out some other way of meeting her. Also, Mary may be unsocial.

Explanation 3: the party interfered with something else she wanted to do.

Response: without knowing what Mary's other plans were, there's not much John can do here. If he knew, then he could expand those plans, looking for an explanation.

Explanation 4: Mary is avoiding some unpleasant state associated with going to the party; i.e., she thought about the party and decided against it.

Response: John expands the plan for going to the party, D-PROX + $PARTY, looking for potential negative states.

D-PROX: again, John decides that Mary would use her car.

1. Problem: she wants to go to the party, but she doesn't want to use her car.

   Explanation 1: Mary doesn't like driving long distances [assuming that John thinks that it's along way from Mary's house to the party].

   Response: none.

   Note: Next time, try to arrange a meeting closer to Mary.

   Explanation 2: she doesn't want to spend the money on gas.

   Response: none.

   Note: Mary may be a tightwad.

John now looks the negative aspects of $PARTY. He expands that into its three components, EAT, DRINK and SOCIALIZE.

1. Problem: Mary wants to go the party, but she wants to avoid eating.

   Explanation: She's on a diet.

   Response: none.

2. Problem: she want to avoid drinking.

   Explanation 1: she doesn't drink at all.

   Response: none.

   Note: She may not be much fun after all.

   Explanation 2: she's a recovering alcoholic.

   Response: none.

   Note: John consults his attitudes towards alcoholics.

3. Problem: she doesn't want to talk to the people at the party.

   Response: expand that into two categories, John and all the others.

   1. Problem: she doesn't want to talk to me.

      Explanation: she doesn't think I'm

attractive, interesting, etc., etc.

Response: I will have to persuade her otherwise. [This could lead to several minutes' worth of exploration.]

2. Problem: she doesn't want to talk to someone else at the party.

   Explanation: Since John knows very little about Mary's attitudes towards the other people at the party, there's not much he can do here.

[End of SOCIALIZE path]

[End of D-PROX + $PARTY path]

4. Problem: Mary got the invitation, but she didn't understand the details.

   Explanation: She got the time or the date wrong.

   Response: Call her up and tell her.

   Note: Don't do that -- it would be socially risky.

5. Problem: She didn't get the invitation.

   Explanation: The Post Office is slow.

   Response: none.

   Note: this plan may work at some later date.

6. Problem: Jane didn't invite Mary at all.

   Explanation 1: Doing so would have interfered with some plan of Jane's.

   Response: Expand Jane's plans for the party.

   Etc.

4.0 Categories of feedback knowledge

What information will need to be included in the system that reasons in this fashion? Obviously, it will need to know both how to generate and how to understand plans, and many existing systems will be useful here.

Wilensky has recently suggested [18] that planning knowledge is one, unified body of information, accessible either through plan-making or plan-understanding. Our own belief is that top-down (planning) information is much more detailed than bottom-up (plan-understanding) information, and our proposed system achieves the same effect by alternating freely between the two. This is, of course, an open question that we do not attempt to resolve here, but we think it is nonetheless important to develop and test such a mechanism.

In particular, there are two components of a feedback simulator that do not arise in separate understanding or synthesis systems: (1) explaining plan failure, and (2) responding accordingly. The following is a partial list of sources of plan failures and possible responses to them.

1. Reason: BAD TIMING. The desired state (goal) is not true at this time; i.e., it may be too early, or it may be too late.

   Response: In the case of something that has not yet happened might be to wait for it to happen or to cause the state to become true sooner. If it's already too late, it may be possible to use the same plan to achieve the same goal, starting over with all the preconditions, etc.

2. Reason: ACCIDENTAL MOTOR SKILL FAILURE, such as missing a golf shot. Deliberate errors of this type are categorized under interference (see below).

   Response: this depends on how likely it was that the plan could have been executed successfully. (One hesitates to call a football game a 2-hour-long accident simply because of motor skill failures.)

3. Reason: ACCIDENTAL SYSTEM FAILURE, such as delayed delivery of mail or a disabled vehicle.

   Response: this is system-specific and may involve system repair if the plan is to be pursued, or excuses about forces beyond our control otherwise. Physical illness is included here.

4. Reason: BELIEF-SPACE INCONSISTENCY. In our sample story, John thought Jane and Mary were friends, but that may have been incorrect and would account for the failure of his plan.

   Response: attempt to reconcile the inconsistency, if possible.

5. Reason: FAULTY CAUSALITY. The planner didn't understand how something "works," such as the rules in a game, so that the situation he expected to occur actually had no basis in reality.

   Response: theplanning rules can be revised, but there is probably little that can be salvaged from the original plan.

6. Reason: TOO-SHALLOW PLANNING. The planner missed some detail that crucially affected the outcome of the plan. This can result from faulty "indices" or plan filters.

   Response: incorporate the missed detail into future indices.

7. Reason: NEGLECTED PRECONDITION. A precondition was not considered during the formation of the plan.

   Response: if the precondition can now be achieved, the plan may yet be salvaged.

8. Reason: FAILED PRECONDITION. A precondition that was asuumed not to be a problem has failed.

   Response: examine the details of the plan for

achieving the precondition and "recursively" to apply the diagnosis procedure to that.

9. Reason: DELIBERATE INTERFERENCE, when another actor has sabotaged the plan.

   Response: simulate that person's planning to attempt to establish the reason for the interference.

10. NON-COMPLIANCE, when a participant in the plan doesn't behave as expected.

    Response: figure out why that person did not comply. The plan in question may have conflicted with that person's goals, there may be a misunderstanding that can be corrected, or there may have been a lengthy evaluation of the pros and cons (which the planner may then try to simulate).

11. Finally, there is always BAD LUCK, specially when the plan involves chance (e.g., dice) or unpredictability (e.g., the weather).

    Response: try, try again?

5.0 Understanding and planning

5.1 Understanding as a technique for planning

Planning in a world in which there is any sort of causality requires being able to compute the consequences of an action, equivalent to a simulation of that action. In robot planning, all the consequences are known with certainty, and the only way to introduce errors into this process is by neglecting to make some inferences, which might be a reasonable byproduct of a system trying to avoid inferential explosion. In any case, consequences must be "understood" -- that is, they must be linked to other states and acts that the system knows about. If the causal model

is complete and error-free, then "perfect" planning is possible, meaning that there will be no surprises in store, although there may still be difficult problems determining the proper sequence of actions so that subgoals do not annihilate each other.

In a simulation of human actors, all bets are off. While some inferences that the actors will make can always be trusted ("context-free" inferences), others depend on what the actor knows. Instead of a formal causality, we must represent belief spaces: what John thinks, what Mary thinks, what John thinks Mary thinks, etc., in varying degrees of depth and accuracy. These belief spaces contain goals, among other things. When John performs some act, such as asking a favor of Mary, she must "understand" that in terms of her beliefs about John's goals, which may not at all correspond to John's actual beliefs, before she can plan what to do next.

The point is that in order to simulate planning, we must also simulate understanding, using some of the same techniques that PAM and ARTHUR, for example, used to link states to plans and plans to goals. In this view, their programs are characters, Pam and Arthur, who are responding to input, albeit somewhat passively. We must have some representation for separate and shared belief spaces, well integrated into the understander.

5.2 Planning as a technique for understanding

PAM and ARTHUR can generate "top-down inferences" from goals to plans to actions, but otherwise their capacity as problem solvers is limited. When they have established that a goal is under pursuit, they make only very simple predictions about what will happen next, rather than to take in account large amounts of existing information and apply problem-solving strategies. ARTHUR can later revise its prediction by seeking another way of explaining why something actually does make sense in terms of the previously established goal. But for all its cleverness, it sometimes resembles a rather inattentive listener: rather than try hard to anticipate what's coming up, it is content to follow along, barely, and have everything explained to it. In short, the predictive components of these programs are weak because they lack planning skills.

5.3 Feedback simulation and shallow planning

To incorporate both planning and understanding components, our simulator has the following structure. Picking an arbitrary point in the cycle, suppose John has already chosen a goal to pursue. A number of plans may present themselves. (Of course, this set may not be the same for Mary, or for John at some other time, and the order may vary.) Some of these plans may have preconditions that John remembers, and he will take those into consideration.

He may now decide to work on them first, in "means-ends" style, but that is only one possible way to deal with them (and the only way TALE-SPIN dealt with them). More likely, the preconditions have contraints of time and other resources that may merge with other planned events. If he does choose the means-ends approach, it is because the planning heuristics indicated that that was reasonable in the current context.

We certainly want to leave the option that there will be preconditions that Joe's does not remember now. Otherwise, we fall prey to the Fallacy of the Perfect Planner, where humans resemble the robot planners. In real life, we don't have the time to anticipate everything in advance, much less the ability, and we sometimes must act quickly, at the risk of making an error. This is especially true in speech [10], where we can easily get ourselves into the kind of bind that we would not if we were writing.

Moreover, there are other ways to become aware of preconditions than having them hard-wired to plans, using hypothetical simulation. For example, if I want to see a film, I can simulate the theater-script by imagining that I'm there. In the first scene of that script, I'm handing some money to the ticket seller. (I "see" this, or "construct" it, or "remember" it, or follow a MOP for it, or whatever.) That is a legitimate way of becoming aware of the "have money" precondition. It may happen at planning

time, or halfway to the theater, or not until I'm actually there. The point is that we wnat to allow flexibility in our model.

Carried to an extreme, this approach would allow us drastically to reduce the backwards inferences we need to make. We would consider only the very local or immediate requirements; as an example, if I'm handing money to someone now, them I have to have money now. Hard-wired long-range preconditions do exist, but they are learned. Human behavior is a mix of the efficient and inefficient.

Suppose now that John has chosen a particular plan, and all the preconditions that he has considered are met (in his belief space, anyway). Some act is called for, and John imagines performing that act -- simulating it in the context of his beliefs. A number of consequences occur, some of which he may have anticipated, others of which he may not. In either case, these consequences are feedback for him, to be reconciled with his goal structure. In the trivial case, the consequences exactly match his goals -- he imagines that the plan works perfectly. AI models with strongly connected causality have this characteristic -- since nothing can go wrong (can go wrong, can go wrong, ...), there's no point to this step. A faulty causality is therefore one source of plan failure, as mentioned in the previous section.

The consequences that were not explicitly intended must now be "understood," and much of the PAM/ARTHUR machinery is quite useful here. For those events that can be connected to goals, John can determine whether the goals conflict with the initial goal, and so on, and he can employ various strategies ("meta-plans" [17,18]) to deal with this. Other consequences may be the product of natural forces or other powers beyond John's control, not amenable to negotiation or discussion, so alternative plans may be sought. The only cost here is the time is takes to think all this through, since John is still imagining all this.

Suppose the consequences are satisfactory, and John decides actually to perform the act, and does so. Now the real simulator produces the actual consequences, and they, too, become feedback for John. If they match his predictions, then he can continue with his plan. But they may not. Something may have gone wrong, and the error must be accounted for and dealt with. It may be that the planner has already formulated a plan for coping with the failure. Contingency planning, in fact, is exactly this: the planner explicitly prepares for certain errors in advance. Related to contingency planning is game-tree planning, where the planner hypothesizes several outcomes that are simply possibilities, not errors. A chess player may be prepared to respond to some specific moves by his opponent, and he may also have a contingency plan to deal with unexpected moves (knock over the board). Contingency plans are thus in

service of higher-level goals.

6.0 Other related work

Plan-based understanding programs assume that the actors in the stories are goal-directed. What the programs are doing, then, is monitoring the execution of the actors' plans. It would seem natural to wonder whether this work shares the approach taken by so-called robot planners or automatic plan-synthesizers [13], and if not, whether they should [9]. The answer is no and yes.

No, because the nature of events in stories differs from that of the automatic planner. In the world of automatic planning, there is usually only one actor, the planner, who has complete knowledge about the world and is faced with the task of figuring out how to achieve some goal, getting it right the first time. Typically, this involves looking at subgoals, their preconditions, and their side-effects. Finding the proper sequence of events, so that achieving one goal doesn't annihilate another, for instance, is a concern of many of these systems. Even those few systems that handle errors "after the fact," like Sussman's HACKER [16], still live in a world of one actor, where time can be reversed by backtracking or undoing. All of these programs struggle to "do it right" the first time, and become very unhappy otherwise.

In contrast, a world of several human actors lacks complete information. Time cannot be reversed, and each actor's belief space may differ in some respects from every other actor's. In fact, one character's belief space will have to contain models of the other actors' belief spaces, which may or may not be consistent with those actors' own belief spaces.

But yes, plan-understanding programs should have the ability to monitor the execution of plans, so that they can diagnose errors and respond appropriately, perhaps taking remedial action.

7.0 Summary

We have examined some of the intricacies of planning in a world of many actors and incomplete information. We have attempted to point out that the two processes of bottom-up plan inference and top-down plan synthesis are inseparable, whether solving a problem or understanding the progress of someone else who is solving a problem. These processes require the use of separate belief spaces (e.g., what John thinks Mary thinks about John). Failures in plan execution are categorized as to source, and as feedback to the planner, they may invoke various responses and suggest future goals to be pursued. We are currently incorporating these ideas in REACTOR, a natural-language understanding system.

ACKNOWLEDGEMENT. Most of the ideas in this paper benefited from lengthy discussions with Rick Granger, who also made valuable suggestions for the various drafts.

REFERENCES

1] American Association for Artificial Intelligence.
Proceedings of the First Annual National Conference on Artificial Intelligence.
Stanford University, August 1980.

2] Robert de Beaugrande and Benjamin N. Colby.
Narrative models of action and interaction.
Cognitive Science 3(1):43-66, January-March 1979.

3] Richard H. Granger.
Adaptive Understanding: Correcting Erroneous Inferences.
PhD dissertation, Yale University, 1980.
Research Report 171, Yale Computer Science Department.

4] Richard H. Granger.
When expectation fails: towards a self-correcting inference system.
in AAAI-1 [1], pages 301-305.

5] Richard H. Granger and James R. Meehan.
Identification of memory processes underlying text understanding: a research proposal.
Technical Report 163, Artificial Intelligence Project, Department of Information and Computer Science, University of California, Irvine, 1980.

6] William C. Mann.
Dialogue games.
Report ISI/RR-79-77, Information Sciences Institute, University of Southern California, October 1979.

7] William C. Mann.
Toward a speech act theory for natural language processing.
Report ISI/RR-79-75, Information Sciences Institute, University of Southern California, March 1980.

8] William C. Mann and James A. Moore.
Computer as author -- results and prospects.
Report ISI/RR-79-82, Information Sciences Institute,
University of Southern California, January 1980.

9] Drew V. McDermott.
Planning and acting.
Cognitive Science 2(2):71-109, April-June, 1978.

10] James R. Meehan.
Future directions in natural language generation.
Technical Report, Department of Information and
Computer Science, UCI (to appear).

11] James R. Meehan.
The Metanovel: Writing Stories by Computer.
PhD dissertation, Yale University, 1976.
Garland Publishing, New York, 1979.

12] James R. Meehan.
TALE-SPIN and Micro TALE-SPIN.
In Schank and Riesbeck [15], 197-258.

13] Earl D. Sacerdoti,
Problem solving tactics.
Technical Note 189, SRI International, July 1979.

14] Roger C. Schank and Robert P. Abelson.
Scripts, Plans, Goals, and Understanding.
Lawrence Erlbaum, Hillsdale, New Jersey, 1977.

15] Roger C. Schank and Christopher K. Riesbeck, editors.
Inside Computer Understanding.
Lawrence Erlbaum, Hillsdale, New Jersey, 1981.

16] Gerald Jay Sussman.
A Computer Model of Skill Acquisition.
PhD dissertation, MIT, 1973.
American Elsevier, New York, 1975.

17] Robert Wilensky.
Meta-planning.
In AAAI-1 [1], 334-336.

18] Robert Wilensky.
Meta-planning: Representing and using knowledge about
planning in problem solving and natural language
understanding.
Memorandum No. UCB/ERL M80/33, 5 August 1980.

19] Robert Wilensky.
Pam and Micro PAM.
In Schank and Riesbeck [15], 136-198.

20] Robert Wilensky.
Understanding Goal-Based Stories.
PhD dissertation, Yale University, 1978.
Research Report 140, Yale Computer Science Department.
Garland Publishing Co., New York, 1979.